

## Geostat Summer School



# QGIS

An Open Source Desktop GIS



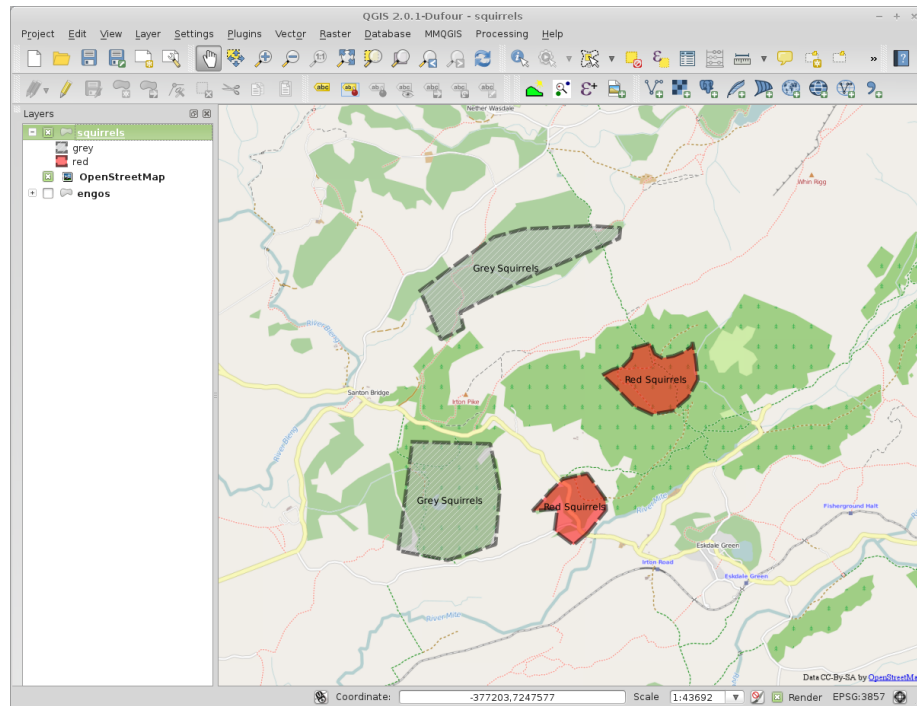
## Intro

### Features

- Cross-platform Lin/Win/Mac Desktop GIS
- Works with Geospatial Standards
- Written in C++
- Embedded Python

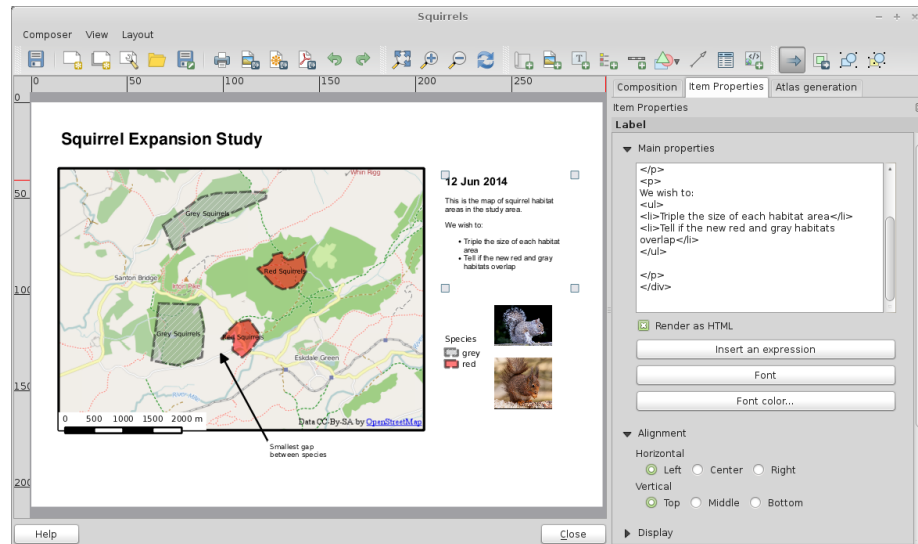
## Screenshot

## Typical screen



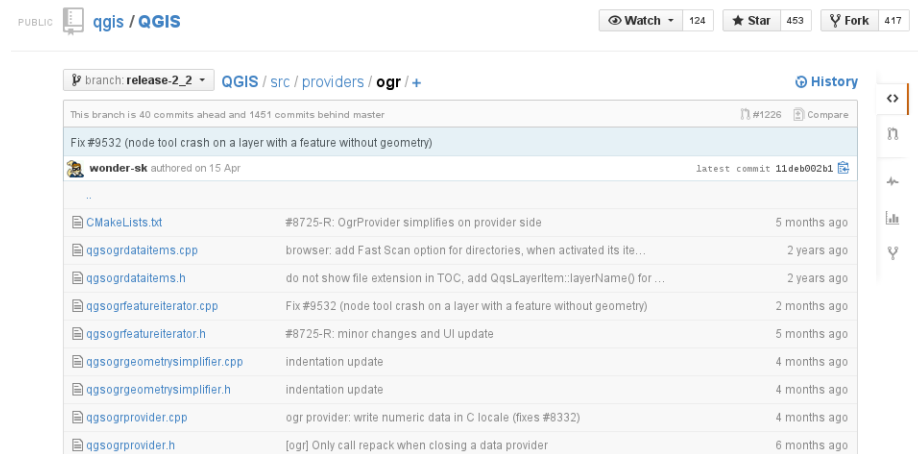
# Composer

## Map Composer



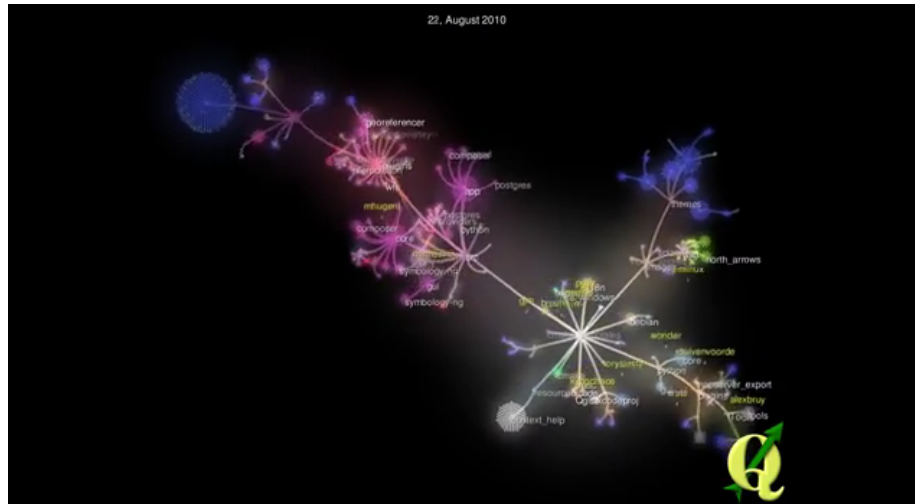
## C++

## C++



## Development

## Visualisation



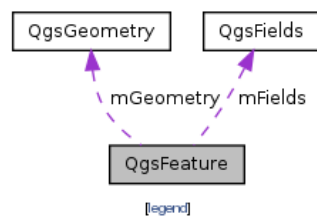
## Feature

## Structure

The feature class encapsulates a single feature including its id, geometry and a list of field/values attributes. [More...](#)

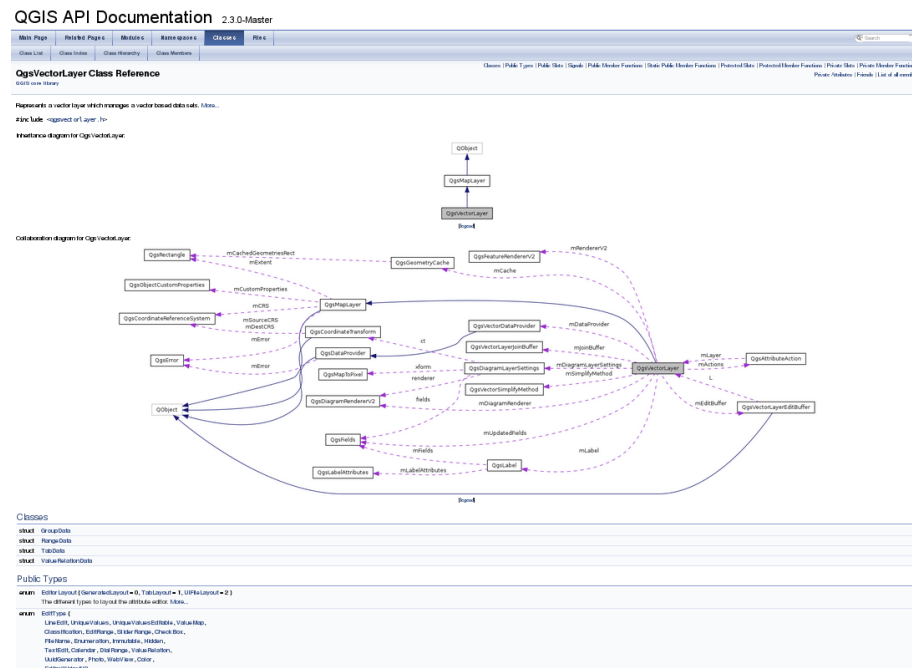
```
#include <qgsfeature.h>
```

Collaboration diagram for QgsFeature:



## Vector Layer

## Structure



This is the structure of a vector layer

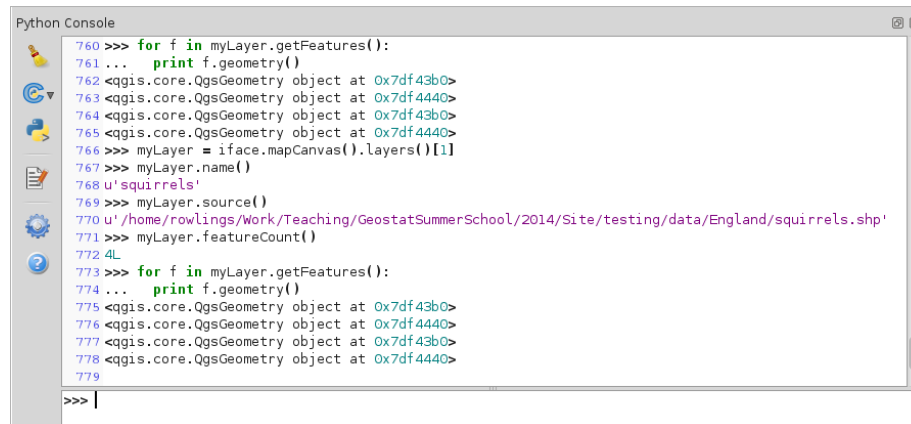
## Feature methods

## Method list

	Set the validity of the feature. <a href="#">More...</a>
<b>QgsGeometry *</b>	<b>geometry ()</b> const Get the geometry object associated with this feature. <a href="#">More...</a>
<b>QgsGeometry *</b>	<b>geometryAndOwnership ()</b> Get the geometry object associated with this feature The caller assumes responsibility for the QgsGeometry's destruction. <a href="#">More...</a>
<b>void</b>	<b>setGeometry (const QgsGeometry &amp;geom)</b> Set this feature's geometry from another <b>QgsGeometry</b> object (deep copy) <a href="#">More...</a>
<b>void</b>	<b>setGeometry (QgsGeometry *geom)</b> Set this feature's geometry (takes geometry ownership) <a href="#">More...</a>
<b>void</b>	<b>setGeometryAndOwnership (unsigned char *geom, size_t length)</b> Set this feature's geometry from WKB. <a href="#">More...</a>
<b>void</b>	<b>setFields (const QgsFields *fields, bool initAttributes=false)</b> Assign a field map with the feature to allow attribute access by attribute name. <a href="#">More...</a>
<b>const QgsFields *</b>	<b>fields ()</b> const Get associated field map. <a href="#">More...</a>
<b>bool</b>	<b>setAttribute (const QString &amp;name, QVariant value)</b> Insert a value into attribute. <a href="#">More...</a>

## Python Console

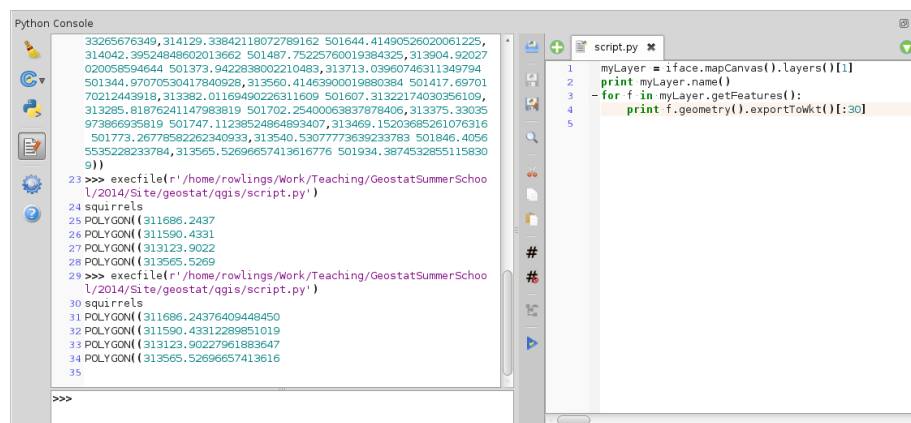
### Console



```
760 >>> for f in myLayer.getFeatures():
761 ...     print f.geometry()
762 <qgis.core.QgsGeometry object at 0x7df43b0>
763 <qgis.core.QgsGeometry object at 0x7df4440>
764 <qgis.core.QgsGeometry object at 0x7df43b0>
765 <qgis.core.QgsGeometry object at 0x7df4440>
766 >>> myLayer = iface.mapCanvas().layers()[1]
767 >>> myLayer.name()
768 u'squirrels'
769 >>> myLayer.source()
770 u'/home/rowlings/Work/Teaching/GeostatSummerSchool/2014/Site/testing/data/England/squirrels.shp'
771 >>> myLayer.featureCount()
772 4L
773 >>> for f in myLayer.getFeatures():
774 ...     print f.geometry()
775 <qgis.core.QgsGeometry object at 0x7df43b0>
776 <qgis.core.QgsGeometry object at 0x7df4440>
777 <qgis.core.QgsGeometry object at 0x7df43b0>
778 <qgis.core.QgsGeometry object at 0x7df4440>
779
>>> |
```

## Python Editor

### Console with Editor



```
1 myLayer = iface.mapCanvas().layers()[1]
2 print myLayer.name()
3 for f in myLayer.getFeatures():
4     print f.geometry().exportToWkt()[0:30]
5
```

```
33265676349,314129.33842118072789162 501644.41490526020061225,
314042.39524848602013662 501487.75225760019384325,313904.92027
020058594644 501373.9422838002210483,313713.03960746311349794
501344.97070530417840928,313560.41463900019880384 501417.69701
70212443919,313982.01169490226311609 501607.31322174030356109,
313285.81876241147983819 501702.25400063837878406,313375.33035
973866935819 501747.11238524864893407,313469.15203685261076316
501773.26778582262340933,313540.53077773639233783 501846.4056
5535228233784,313565.52696657413616776 501934.3874532855115830
9))
23 >>> execfile(r'/home/rowlings/Work/Teaching/GeostatSummerSchoo
l/2014/Site/geostat/qgis/script.py')
24 squirrels
25 POLYGON ((311686.2437
26 POLYGON ((311590.4331
27 POLYGON ((313123.9022
28 POLYGON ((313565.5269
29 >>> execfile(r'/home/rowlings/Work/Teaching/GeostatSummerSchoo
l/2014/Site/geostat/qgis/script.py')
30 squirrels
31 POLYGON ((311686.24376409448450
32 POLYGON ((311590.43312289851019
33 POLYGON ((313123.90227961883647
34 POLYGON ((313565.52696657413616
35
>>>
```

## Squirrels

Red and Grey







CC-licensed from flickr users imh and hisgett

## Squirrels

Red and Grey

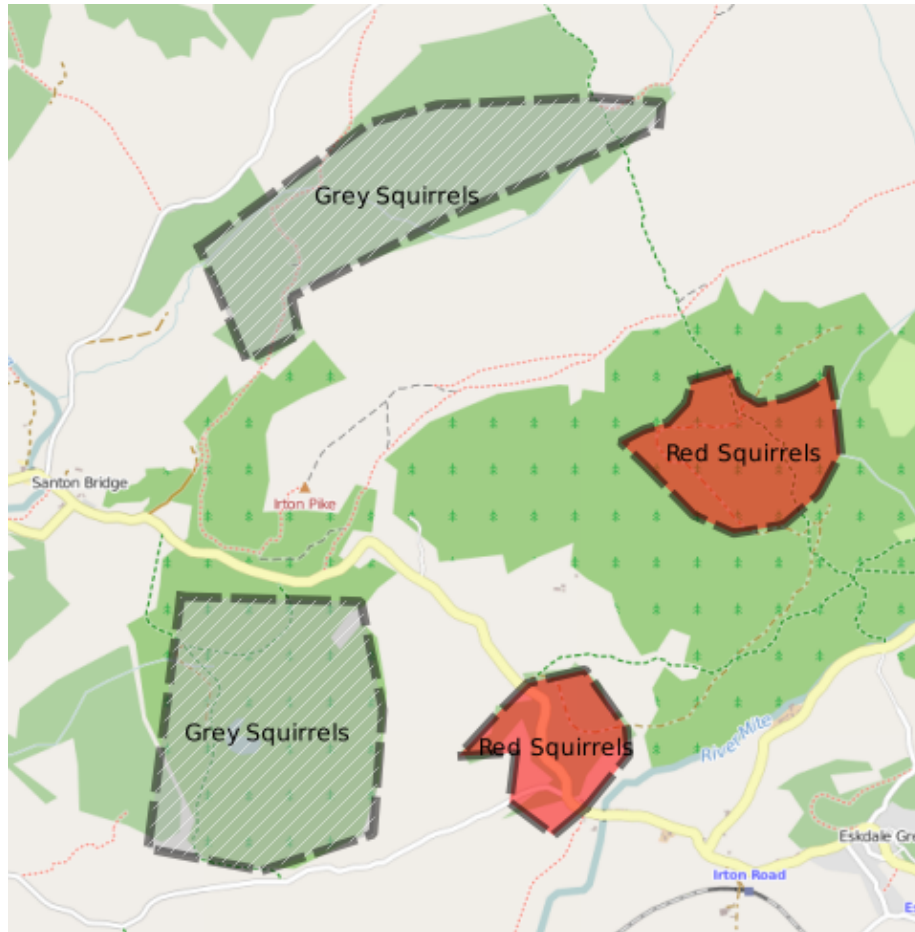




CC-licensed from flickr users king-edward and hisgett

## Status

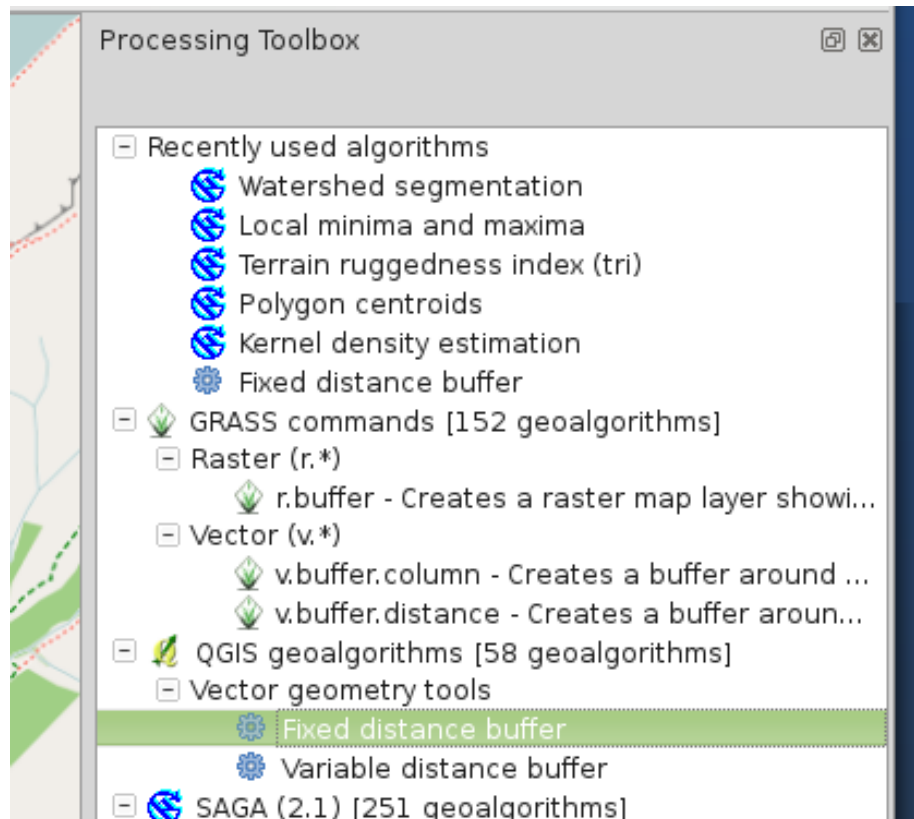
## Distribution



- What if populations triple and areas triple?
- Can we buffer the regions and check for overlap?

## Buffering

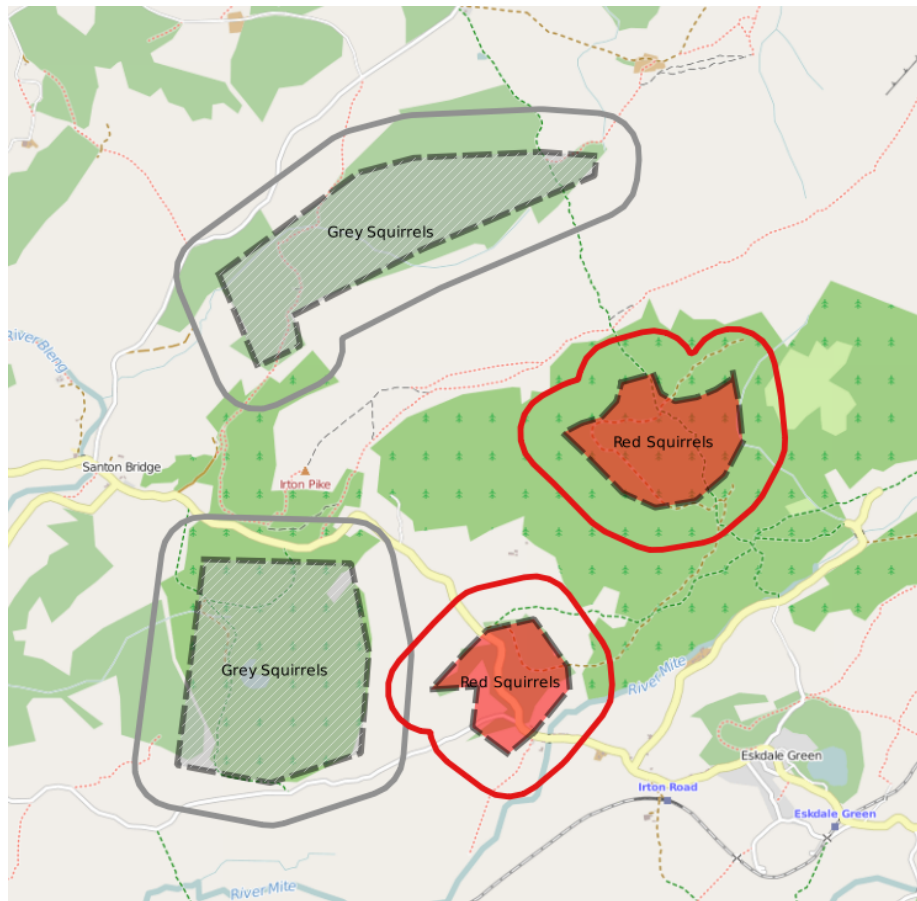
### Options



### Fixed Buffer

Styled...





- How do we find a buffer width that scales an area?

## Strategy

### Binary Search

- Area of buffer increases as width increases
- Use a binary search algorithm to find  $w$  such that area of buffered region =  $S * \text{area of region}$

## QGIS Processing Scripts

### Easy

```
##[BSR scripts]=group
##areas=vector
##scale_factor=number 2
##buffered=output vector

from PyQt4.QtCore import *
from qgis.core import *

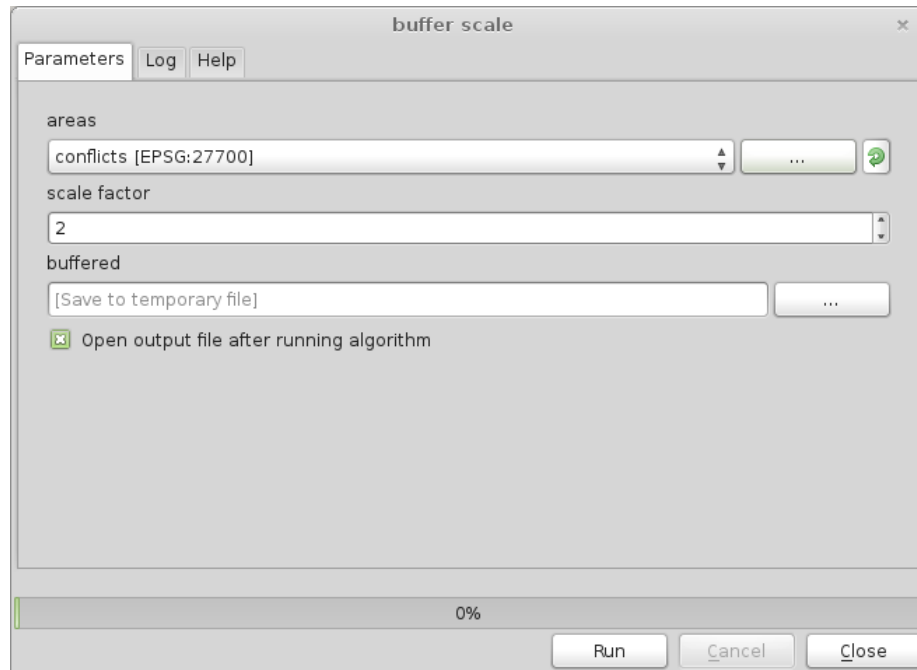
import processing
from processing.core.VectorWriter import VectorWriter
from bsrutils import rescaleBuffer

areas_layer = processing.getobject(areas)
areas_features = processing.getfeatures(areas_layer)
fields = areas_layer.pendingFields().toList()
writer = VectorWriter(buffered, None, fields, POLYGON, areas_layer.crs() )

for feature in areas_features:
    outFeat = rescaleBuffer(feature, scale_factor)
    writer.addFeature(outFeat)
```

ui

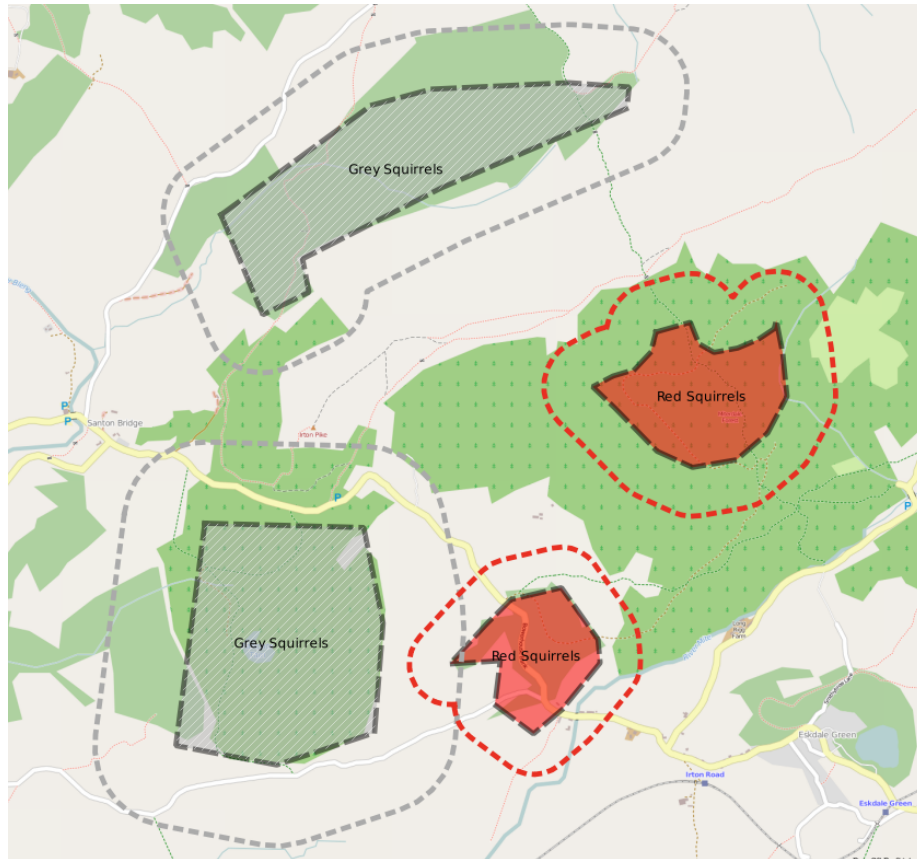
Gets this





Run

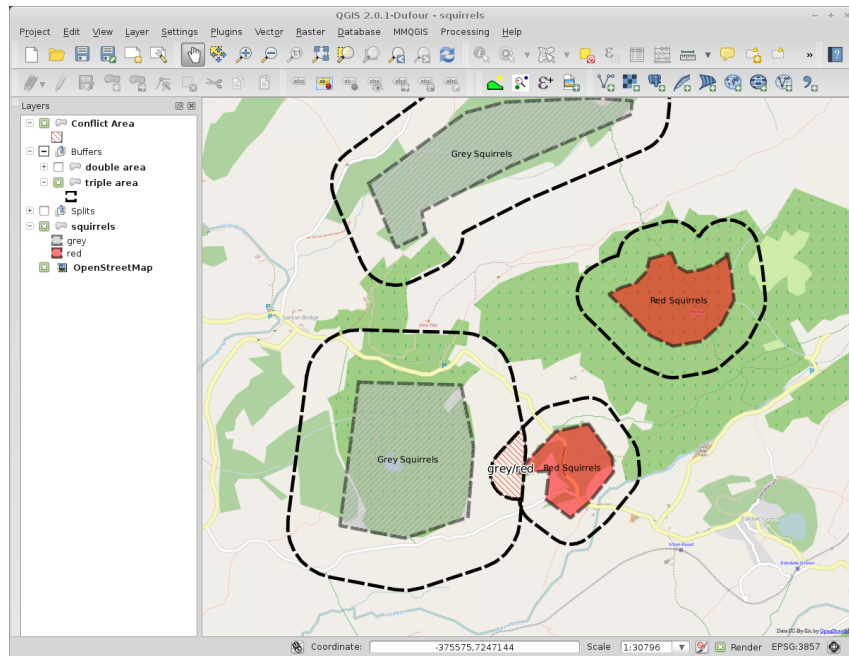
and get



## Conflict Area

### Getting the conflict area

- Split layer on attribute value
- Intersection of those layers



- 
- Can we automate this?

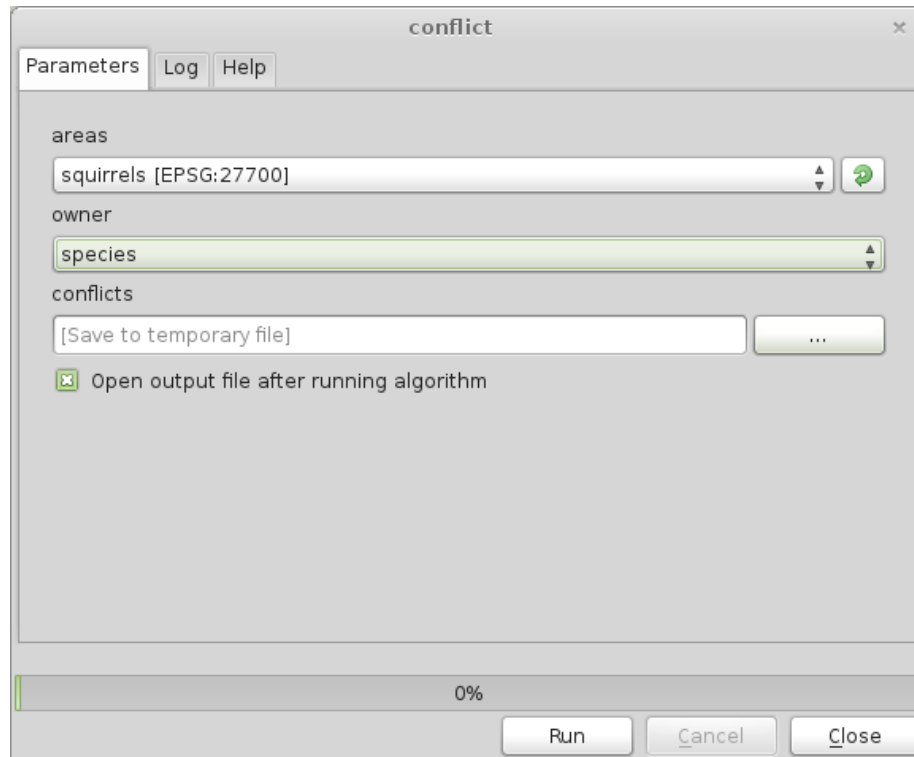
## Conflict processing script

### outline

```
##[BSR scripts]=group
##areas=vector
##owner=field areas
##conflicts=output vector
...
for aFeature in features:
    for bFeature in features:
        if ownerValue(aFeature) != ownerValue(bFeature):
            if aFeature.geometry().intersects(bFeature.geometry()):
                outFeature.setGeometry(aFeature.geometry().intersection(bFeature.geometry()))
                writer.addFeature(outFeature)
```

## Conflict UI

### Dialog



### Two steps

#### Can we do better?

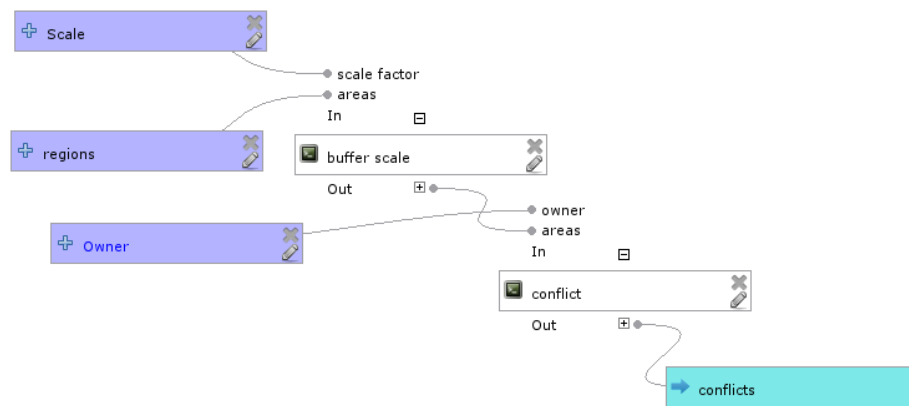
- Buffer areas by scale factor
- Compute conflict areas

### Models

- Allows connected algorithms
- Can use any algorithms in the processing toolbox
- Graphical tool connecting inputs, algorithms, outputs

## Model builder

### Expansion Model



## Model builder

### Expansion UI

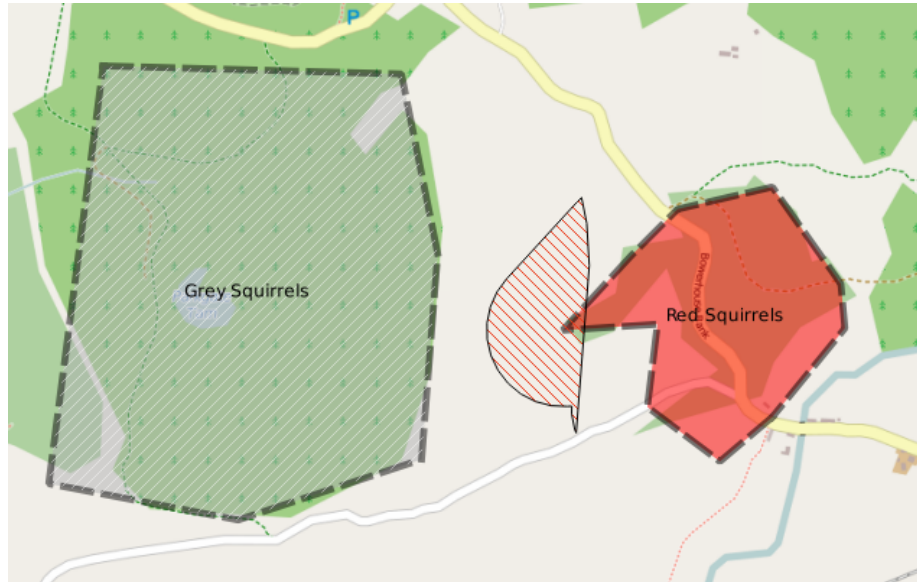
The screenshot shows the **Expansion** dialog box with the following settings:

- regions**: squirrels [EPSG:27700]
- Owner**: species
- Scale**: 3
- conflicts**: [Save to temporary file]
- ☒ Open output file after running algorithm

The progress bar at the bottom indicates 0% completion. The **Run**, **Cancel**, and **Close** buttons are visible at the bottom right.

## Conflict Output

One step, one extra layer



## Other advanced QGIS features

### Good stuff

#### Python Plugins

- Build Custom GUIs
- Create Menus
- Define New Layer Types
- Define New Renderers

#### R Processing Scripts

- Leverage R-spatial
- Examples Included

#### Feature Editing/Cleanup

#### Spatial Database Integration

#### OGC Services